TravelMatch

# User Requirements Document

Version 1.2.1

D.J. van den Brand (0772180)
S. He (0810831)
J.M.A.P. van Hoof (0778486)
G.C. Linders (0815449)
M.J.M. Muijsers (0805654)
G.H. Santegoeds (0890429)
L.D. Stooker (0819041)
J.W.J.H. Visser (0828234)

22nd June, 2015

**Abstract**

This document contains the user requirements for the TravelMatch application, which is used to help people find their holiday destination. This application is developed in the Software Engineering Project at Eindhoven University of Technology. The user requirements in this document are defined in consultation with the customer, Menno Veen, representing iLysian B.V. This document complies with the ESA software standard [1].

# Contents

**Document Status Sheet** **3**
General . . . 3
Document history . . . 3

**Document Change Records** **4**
General . . . 4
Changes . . . 4

**1 Introduction** **5**
1.1 Purpose . . . 5
1.2 Scope . . . 5
1.3 Definitions and abbreviations . . . 5
1.3.1 Definitions . . . 5
1.3.2 Abbreviations . . . 6
1.4 References . . . 6
1.5 Overview . . . 8

**2 General description** **9**
2.1 Product perspective . . . 9
2.2 General capabilities . . . 9
2.2.1 Interest analysis . . . 9
2.2.2 AI module . . . 9
2.2.3 Hotel overview . . . 10
2.3 General constraints . . . 10
2.4 User characteristics . . . 10
2.4.1 Users . . . 10
2.4.2 Managers . . . 11
2.4.3 Administrators . . . 11
2.4.4 Developers . . . 11
2.5 Environment description . . . 11
2.6 Assumptions and dependencies . . . 12

**3 Specific requirements** **13**
3.1 Capability requirements . . . 13
3.1.1 General . . . 13
3.1.2 Registration screen . . . 15
3.1.3 Login screen . . . 16
3.1.4 User details screen . . . 16
3.1.5 About screen . . . 17
3.1.6 Vacation details . . . 17
3.1.7 Interest analysis . . . 18
3.1.8 Hotel overview . . . 19
3.1.9 Hotel details . . . 21
3.1.10 Back end . . . 21
3.1.11 Analytics . . . 24
3.2 Constraint requirements . . . 25
3.2.1 App environment . . . 25
3.2.2 Adaptability . . . 26

3.2.3 Resources . . . 26
3.2.4 Licensing . . . 26
3.2.5 Performance Requirements . . . 26
3.3 Changes in user requirements . . . 28

**A Use cases** **30**
A.1 General use cases . . . 30
A.1.1 Register an account via email . . . 30
A.1.2 Log into an account via email . . . 30
A.1.3 Register and log into an account via Facebook . . . 31
A.2 Specific use cases . . . 32
A.2.1 Trying out without creating an account . . . 32
A.2.2 Booking a vacation on a new device . . . 32

# Document Status Sheet

## General

| | | |
|---|---|---|
| **Document title:** | User Requirements Document | |
| **Document identifier:** | TravelMatch.Doc.URD/1.2.1 | |
| **Authors:** | D.J. van den Brand | (0772180) |
| | S. He | (0810831) |
| | J.M.A.P. van Hoof | (0778486) |
| | G.C. Linders | (0815449) |
| | M.J.M. Muijsers | (0805654) |
| | G.H. Santegoeds | (0890429) |
| | L.D. Stooker | (0819041) |
| | J.W.J.H. Visser | (0828234) |
| **Document status:** | Final document | |

## Document history

| Version | Date | Author(s) | Reason |
|---|---|---|---|
| 0.1 | 21-04-2015 | G.C. Linders | Initial version. |
| 0.2 | 24-04-2015 | D.J. van den Brand, G.C. Linders, G.H. Santegoeds | First incomplete draft. |
| 0.3 | 29-04-2015 | D.J. van den Brand, G.C. Linders, G.H. Santegoeds | Second draft. |
| 0.4 | 30-04-2015 | D.J. van den Brand, G.C. Linders, G.H. Santegoeds | Third draft for client approval. |
| 0.5 | 01-05-2015 | D.J. van den Brand, G.C. Linders, G.H. Santegoeds | Fourth draft for client approval. |
| 0.6 | 04-05-2015 | D.J. van den Brand, G.C. Linders, G.H. Santegoeds | Fifth draft for client approval. |
| 1.0 | 06-05-2015 | D.J. van den Brand, G.C. Linders, G.H. Santegoeds | Final version for client approval. |
| 1.0.1 | 06-05-2015 | D.J. van den Brand, G.C. Linders, G.H. Santegoeds | Small spelling and typo fixes. |
| 1.0.2 | 06-05-2015 | G.C. Linders | Clarified domain model. |
| 1.1 | 27-05-2015 | G.H. Santegoeds | Client decided on a new search model. |
| 1.2 | 04-06-2015 | G.H. Santegoeds | Client switched to specific feed of previous provider. |
| 1.2.1 | 12-06-2015 | G.H. Santegoeds, L.D. Stooker, D.J. van den Brand | UR changes agreed with client. |

# Document Change Records

## General

**Date:** 22nd June, 2015
**Document title:** User Requirements Document
**Document identifier:** TravelMatch.Doc.URD/1.2.1

## Changes

| Version | Date | Section | Reason |
|---|---|---|---|
| 1.1 | 27-05-2015 | All throughout | Updated mentions of affiliate networks to include the newly adopted usage of Waverunner and optional usage of Daisycon/Tradetracker. |
| 1.2 | 04-06-2015 | All throughout | Changed from Waverunner to the new usage of solely the Arke feed.<br>Redo of specific requirements to reflect latest client wishes. |
| 1.2.1 | 12-06-2015 | All throughout | Small UR changes to reflect customer wishes for the delivery version. |

# Chapter 1
# Introduction

## 1.1 Purpose

This User Requirements Document (URD) contains the requirements for TravelMatch. These requirements are a negotiated agreement between the client, iLysian B.V., and the TravelMatch development team. All listed requirements, and only these, will be implemented in TravelMatch according to their respective priorities. Any further changes require the full consent of both parties.

## 1.2 Scope

TravelMatch is an application designed for smartphones and tablets, conceived by iLysian B.V. and developed by the TravelMatch development team. The purpose of the application is to assist users in planning a vacation by showing them images from various destinations and hotels or other places to stay. The application employs machine learning to build a profile of the user in order to suggest the ideal trip.

## 1.3 Definitions and abbreviations

### 1.3.1 Definitions

| | |
|---|---|
| Affiliate Network | A network that enables you to receive money from customer redirection [18] |
| Analytics Data | The log of analytics events that is recorded and stored on the database. |
| Android | A popular open-source operating system for embedded devices, including smartphones and tablets, created by Google. |
| Angular JS | An open-source web application framework maintained by Google. |
| Cosine similarity | A measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them. |
| Destination advice | The city, and selection of hotels, that is advised to a user after performing one or more interest analyses. |
| *Destination attributes* or tags | Each destination will have one or more *destination attributes* with an associated numerical relative value, those attributes cover the same preferences as the *DNA attribute*. |
| *DNA attribute* or tags | These are the attributes that the client wants to use to compose the DNA of. In the beginning 10 attributes are chosen and each image shall have a relative numerical value on one or more of the attributes. Attributes can be added or removed later for new and existing images and DNA. |
| Google Play Store | A public repository of free and paid apps for Android, managed by Google. |
| Guest user | An user that does not provide login details but still uses the TravelMatch app. |
| Hotelstars rating | A hotel classification with common criteria and procedures in participating countries to rate a hotel's quality. See [21]. |

| | |
|---|---|
| iLysian | Short for iLysian B.V., a software engineering company situated in Eindhoven, Netherlands. The client for the TravelMatch project. |
| Interest analysis | The action the user will do of judging the images. |
| iOS | A popular closed-source operating system for smartphones and tablets created by Apple. |
| iOS App Store | A public repository of free and paid apps for iOS, managed by Apple. |
| JWT | JSON Web Token: a compact URL-safe means of representing claims to be transferred between two parties, and used in TravelMatch as authentication token, since it is self-validating. |
| Relational database management system (RDBMS) | A database management system (a piece of computer software that interacts with users, other applications and a database to capture and analyze data) based on the relational model (commonly based on the relational database model) |
| TCP/IP | A computer networking model and set of communication protocols used on the internet and similar computer networks, including the Transmission Control Protocol (TCP) and the Internet Protocol (IP) |
| Tinder | A popular dating application for smartphones and tablets featuring a swipe based interface, where a swipe to the left indicates a dislike and a swipe to the right indicates a like. |
| *Travel DNA* | A collection of information about vacation preferences of a specific user or, more specifically, one vacation of that user. This information is stored on the server in a table with values representing the respective gain per attribute for each image the user has swiped. |
| TravelMatch | An application for smartphones and tablets that assists users in planning a vacation. The subject of this project. |
| TravelMatch team | A team of Computer Science students at Eindhoven University of Technology who will design and implement the TravelMatch application. |
| User | The user of the app. |
| *Waverunner* | Waverunner Search Service by Pyton Communication Services; a search service that provides vacation offers and prices of participating travel agencies. |

### 1.3.2 Abbreviations

| | |
|---|---|
| AI | Artificial Intelligence |
| APK | Android Application Package |
| App | Application for smartphones and tablets |
| CMS | Content Management System |
| ESA | European Space Agency |
| IPA | iOS App Store Package |
| OS | Operating System |
| TU/e | Eindhoven University of Technology |
| URD | User Requirements Document |

## 1.4 References

[1] ESA PSS-05-0 Issue 2, Software requirements and architecture engineering process, February 1991

[2] TravelMatch team. User Requirement Document. Version 1.2.1. 22 June 2015.

[3] TravelMatch team. Software Requirements Document. Version 1.0. 22 June 2015.

[4] TravelMatch team. Architectural Design Document. Version 1.0. 22 June 2015.

[5] TravelMatch team. Detailed Design Document. Version 1.0. 22 June 2015.

[6] TravelMatch team. Software User Manual. Version 1.0. 22 June 2015.

[7] TravelMatch team. Software Transfer Document. Version 1.0. 22 June 2015.

[8] TravelMatch team. Unit Test Plan. Version 1.0. 22 June 2015.

[9] TravelMatch team. Integration Test Plan. Version 1.0. 22 June 2015.

[10] TravelMatch team. Acceptance Test Plan. Version 1.0.2. 22 June 2015.

[11] TravelMatch team. Software Configuration Management Plan. Version 1.0. 22 June 2015.

[12] TravelMatch team. Software Project Management Plan. Version 1.0. 22 June 2015.

[13] TravelMatch team. Software Quality Assurance Plan. Version 1.0. 22 June 2015.

[14] TravelMatch team. Software Verification and Validation Plan. Version 1.0. 22 June 2015.

[15] Tom Preston-Werner. Semantic Versioning 2.0.0. Retrieved 6 May 2015. http://www.semver.org/

[16] Coley Consulting. MoSCoW Prioritisation. Retrieved 29 April 2015. http://www.coleyconsulting.co.uk/moscow.htm

[17] Tinder, Inc. Tinder. Retrieved 29 April 2015. http://www.gotinder.com/

[18] Organized Shopping, LLC. Affiliate Network. Marketing Terms. Retrieved 29 April 2015. http://www.marketingterms.com/dictionary/affiliate_network/

[19] Daiycon. About Daisycon. Retrieved 29 April 2015. http://www.daisycon.com/en/about_daisycon/

[20] Drifty Co. Ionic: Advanced HTML5 Hybrid Mobile App Framework. Retrieved 30 April 2015. http://ionicframework.com/

[21] Hotelstars Union. Classification criteria 2015-2020. Retrieved 1 May 2015. http://www.hotelstars.eu/index.php?id=criteria

[22] Django. http://www.django-cms.org/en/

[23] Django administration module. The Django Django admin site. Retrieved 1 June 2015. https://docs.djangoproject.com/en/1.8/ref/contrib/admin/

[24] Django Software Foundation. The Web framework for perfectionists with deadlines — Django. Retrieved 1 June 2015. https://www.djangoproject.com/

[25] Facebook User ID. User IDs and Friends. Retrieved 2 June 2015. https://developers.facebook.com/docs/apps/upgrading#upgrading_v2_0_user_ids

[26] ImageMagick. ImageMagick: Convert, Edit, Or Compose Bitmap Images. Retrieved 2 June 2015. http://www.imagemagick.org/

[27] Google. AngularJS - Superheroic JavaScript MVW Framework. Retrieved 1 June 2015. https://angularjs.org

[28] Adobe Systems Inc. Phonegap: Home. Retrieved 1 June 2015. http://phonegap.com/

[29] Xamarin Inc. Mobile App Development & App Creation Software - Xamarin. Retrieved 1 June 2015. http://xamarin.com/

[30] Eric Raymond. The Jargon File. Version 4.4.7. Retrieved 17 June 2015. http://www.catb.org/jargon/html/

[31] Python Software Foundation. Classes. The Python Tutorial. Retrieved 18 June 2015. https://docs.python.org/2/tutorial/classes.html

[32] Python Software Foundation. PEP 0008 – Style Guide for Python Code. 1 August 2013. https://www.python.org/dev/peps/pep-0008/

[33] Django Software Foundation. Coding style. Retrieved 18 June 2015. https://docs.djangoproject.com/en/1.8/internals/contributing/writing-code/coding-style/

[34] Django Software Foundation. Writing your first Django app, part 1. Database setup. Retrieved 18 June 2015. https://docs.djangoproject.com/en/1.8/intro/tutorial01/#database-setup

[35] Massachusetts Institute of Technology. MIT License. Retrieved 21 June 2015. http://opensource.org/licenses/MIT

[36] Apache Software Foundation. Apache License, Version 2.0. January 2004. http://www.apache.org/licenses/LICENSE-2.0

## 1.5 Overview

The remainder of this document consists of a general description of the TravelMatch application in chapter 2. Sections 2.1 and 2.2 discuss the product perspective and general capabilities respectively. Section 2.3 discusses the general constraints the TravelMatch team must comply with. Section 2.4 describes the different user groups that will be using TravelMatch. Section 2.5 describes the environment in which TravelMatch will operate.

Chapter 3 lists the specific requirements and their respective priorities, divided into logical categories. **Note:** some user requirements have been changed in agreement with the client after the document was accepted. The list of changes is presented in Section 3.3.

# Chapter 2
# General description

## 2.1 Product perspective

At present, many travel agencies expect their customers to know beforehand what their destination is going to be. However, there are groups of people who do not know where they want to go on holiday. The TravelMatch app will help people in choosing their destination through an interest analysis. It then offers some hotels, optionally including a return flight in that destination. Once a suitable destination and hotel have been found, the user is forwarded to the travel agency website to book the trip.

The TravelMatch app can be installed easily using the existing mobile operating system of smartphones and tablets. It is based on the simplicity of Tinder [1] and uses Artificial Intelligence to give people the best possible advice at home. The TravelMatch app features functionality that no travel agency currently provides (for free). TravelMatch is not built on any existing systems. TravelMatch will be developed further by the client when this project is finished.

## 2.2 General capabilities

The TravelMatch app will have the capabilities described below, divided into several categories.

### 2.2.1 Interest analysis

The core feature of the TravelMatch app is the interest analysis. First, the user enters the departure and arrival dates of the vacation, the budget and the total number of adults and children. Then, the user is shown a number of images, and they can choose whether they like or dislike each image. These images show certain characteristics of the holiday that should indicate what the user is looking for in their vacation. They might feature images of an airplane for far away destinations, or images of people hiking for mountainous terrain for adventure. The choices of the user are stored and used by the back end server to build up a *Travel DNA*.

### 2.2.2 AI module

The TravelMatch app is supported by a back end server, which has various components of its own. One of these components is the AI machine learning module. During the interest analysis, the TravelMatch app sends the user's choices to the back end server. The back end server collects these results and produces a so-called *Travel DNA*. Next, this *Travel DNA* is used to find out which destination best suit the user's preferences. The back end server combines these destinations with the vacation details set by the users, and retrieves details for hotels which are available in each destination. Finally, this information is sent back to the TravelMatch app.

Machine learning is employed by the back end server to optimize the results of the AI module. Using analytics gathered from the app and the user's Travel DNA, the parameters of each destination may be adjusted. For instance, if a user whose *Travel DNA* indicates that they love beaches, is recommended a specific destination, then that destination may be recommended to other users who love beaches more frequently.

All destinations in the back end database are initialized with parameters that are set manually by the managers. This is the learning phase of the AI. Once the app is in use by users, these parameters will then be affected by the users' choices.

[1] Tinder is an existing app that uses liking and dislike of profile pictures for dating [17]

### 2.2.3 Hotel overview

Based on the results of the interest analysis, the back end server finds a destination that best suits the user. It then produces a list of hotels that meet the user's needs based on the provided vacation details, and sends this info to the TravelMatch app. The app then displays a list of hotels that are available. Each hotel has at least a picture, a hotel rating (between zero and five stars), a title and a price tag.

In the overview, the user can select a hotel to obtain more information. The additional information includes extra pictures and an expanded description. Also from hereon the user can book the hotel. The user will then be redirected, in their default browser, to the website of a travel agency.

In case the user is not satisfied with the app's advice, they can opt for a second advice. The app will then display the hotels that are available for the second best matching city in accordance with the user's *Travel DNA*. If the user is still not satisfied with the offerings, the user needs to do another interest analysis, altering his *Travel DNA*, before a new 'first advice' is given. This process can be repeated indefinitely. The hotel overview therefore has a total of three different screens that can be requested by the user: to give the first advice, to give the second advice and to show a previously saved advice.

## 2.3 General constraints

Many users can potentially use the app at the same time. To give the users a good result, the set of pictures yet to be judged by the user may change based on the information gained from each prior choice. The server must be able to search for and deliver new photos for every user within a reasonable response time. Therefore, the back end server must be fast, scalable and reliable. As a result, the app requires a constant connection to the Internet (and thus, to the back end server). Also, the app should be responsive enough to the point that it does not feel sluggish or freeze intermittently.

Furthermore, the system is meant to be developed further by iLysian B.V., so the app and the back end server must also be easy to maintain and extend. Initially, the TravelMatch app will support external login via Facebook; in the future, it must be possible to easily add different authentication providers. The TravelMatch app will use the Arke feed to obtain vacation offers at first. It can, however, also function with different feeds from *Daisycon* and *TradeTracker* or switch to a *Waverunner* system at a later time.

In the back end server, it must be possible for a manager without any technical background to manage the database using the CMS.

## 2.4 User characteristics

There are a number of user groups that will be using the TravelMatch app. They are described below.

### 2.4.1 Users

The users download and install the app via the Google Play Store or iOS App Store. They will use the app in order to receive advice for a vacation destination and available hotels. In the group of users there are two subgroups, listed below.

#### Indecisive travelers

The first group consists of users who would like to plan a vacation, but do not know where they wish to go (yet). The needs of these users must be met with the interest analysis feature that selects a destination for the user.

**Budget travelers**

The second target group consists of users who want to plan a vacation, but have a limited budget for doing so. This group must be satisfied by allowing users to select a fixed budget for their vacation. The costs of each vacation in the resulting advice must not exceed this budget.

### 2.4.2 Managers

The back end server for the TravelMatch app requires managers that keep the back end database up-to-date. Managers have read and write access to the back end database and can upload new photos, hotel listings, destinations. They are also able to manage the tags per destination and per photo. The data that is uploaded by the managers is served to end-users through the TravelMatch app. Managers must be able to do all this through a user-friendly interface, without the need for a technical background.

### 2.4.3 Administrators

Administrators are responsible for the back end server and should be able to solve any problems that arise during operation, such as server crashes. Furthermore, they must be able to add or remove managers.

### 2.4.4 Developers

The TravelMatch developers are responsible for maintaining compatibility for future OS releases. They must have unrestricted access to the TravelMatch app and back end server, including all source code. Furthermore, they have access to the back end database as well as the Google Play Store and iOS App Store developer accounts in order to upload new versions of the app.

## 2.5 Environment description

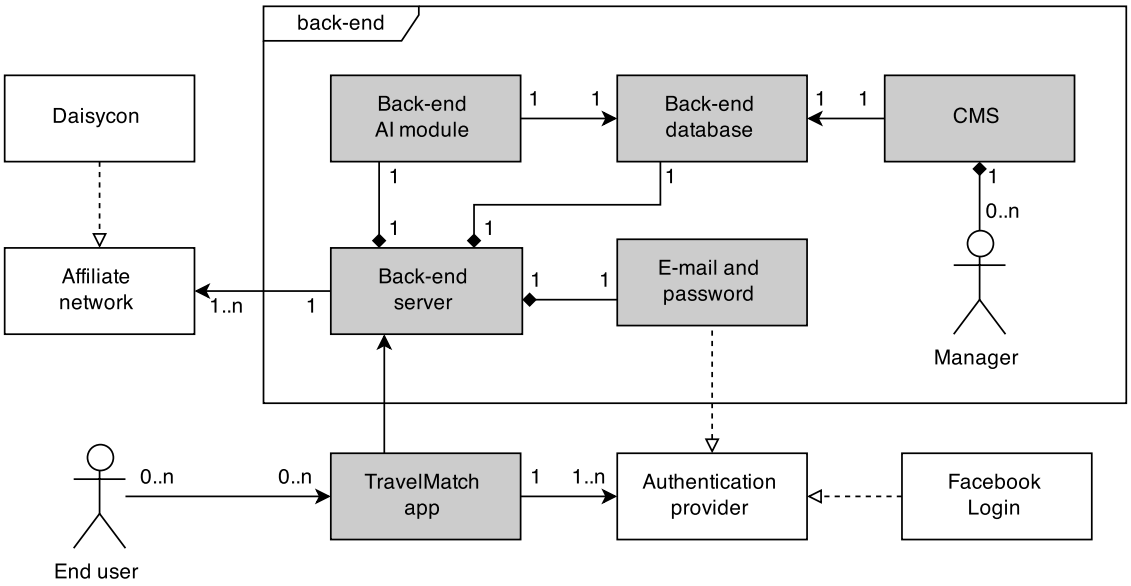


Figure 2.1: Domain Model

The domain model for TravelMatch can be found in Figure 2.1. All components to be implemented by the TravelMatch team are highlighted in gray.

For login via social media, the app communicates with any number of external authentication providers via the Internet. In the current scope, this includes only Facebook and login with an email address and password. For the latter, the app connects to the TravelMatch back end server.

The TravelMatch app can query the back end server. The back end server consists of three submodules, namely the back end AI module, the back end database and the email and password authentication provider. The back end AI module determines the vacation recommendations for the user, based on their *Travel DNA*. It also determines which photos to show the user during interest analysis. As such, it can directly query the back end database without needing to go through the back end server. The back end database has the role of storing data user accounts, photos, tags, vacations and analytics data. Furthermore, a CMS allows for the back end database to be queried and modified. Managers use this CMS to interact with the database.

The server acts as a buffer to store, filter and sort information from the Arke feed before it is passed on to the UI. Additionally, the back end server has the ability to communicate with other feeds instead.

## 2.6 Assumptions and dependencies

In order for the TravelMatch app to function correctly, the following assumptions and dependencies must be met.

- iLysian B.V. will provide data for the back end server, including photos and holiday destinations.
- iLysian B.V. will be responsible for the deployment and distribution of the TravelMatch app.
- The TU/e will provide the hardware for the back end server.
- The server of *TradeTracker* should be up and running with an up to date Arke feed.
- The website of Arke should be online for users to be forwarded and complete their booking.

# Chapter 3
# Specific requirements

In this chapter we state the requirements and constraints of the product. The product will adhere all of these requirements. To prioritize how important these requirements are, we use the MosCoW model [16]. The capital letters in MoSCoW stand for:

**M** *Must have*; these requirements are essential for the product.

**S** *Should have*; these requirements are not critical for the product to work, but are nearly as important as the must haves, meaning they must be implemented if at all possible.

**C** *Could have*; requirements which are not critical to the products success. If they can be implemented with little development costs, they can increase the Clients satisfaction.

**W** *Won't have*; these requirements will not be implemented in this project. However, it would be nice to have them in future versions of the product.

## 3.1 Capability requirements

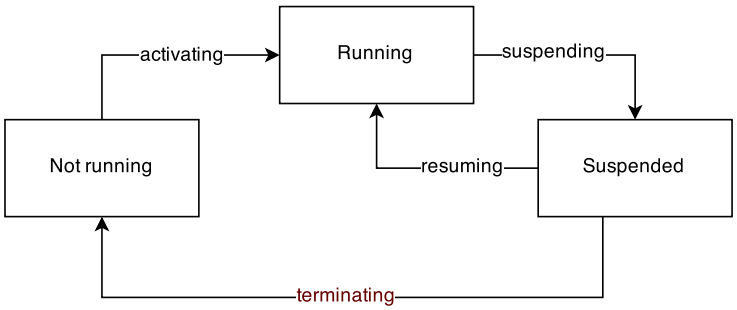


Figure 3.1: Life Cycle

Figure 3.1 gives a simplified overview of the states an app can be in from the OS's perspective. The blocks represent the states and the arrows are the allowed state transitions with their respective names. These states are relevant for some of the following requirements.

### 3.1.1 General

**UCR1** Must have
When activating the app a splash screen is shown while loading.

---

**UCR2** Should have
If no user is authenticated and guest mode is currently inactive on that device, after the splash screen,

a welcome screen is displayed.

**UCR2a** Should have
The welcome screen enables the user to use the app without an account.

**UCR2b** Should have
The welcome screen enables the user to go to the login screen.

**UCR2c** Should have
The welcome screen enables the user to go to the registration screen.

**UCR2d** Should have
The welcome screen enables the user to connect with Facebook.

**UCR3** Must have
When switching to a different application on a user's smartphone, the app is suspended.

**UCR4** Must have
When resuming the app, the app shows the screen that was last displayed the previous time the app was opened.

**UCR5** Should have
When resuming or activating the app, the last logged in user remains logged in without having to enter a password.

**UCR6** Must have
If the user is required to wait while the app is processing or connecting, a loading icon is displayed.

**UCR7** Must have
If a connection error occurs, the app notifies the user.

**UCR8** Must have
On smartphones, the app can function in portrait mode.

**UCR9** Must have
On smartphones, the app blocks landscape mode.

**UCR10** Must have
On tablets, the app can function in portrait mode.

**UCR11** Must have
On tablets, the app can function in landscape mode.

**UCR12** Should have
The app supports switching between multiple languages.

**Sidebar Menu when logged in**

**UCR13** Must have
The sidebar menu enables the user to go to their user details screen.

**UCR14** Should have
The sidebar menu displays previously saved destinations.

**UCR15** Should have
The sidebar menu enables the user to open previously saved destinations.

**UCR16** Should have
The sidebar menu enables the user to delete previously saved destinations.

**UCR17** Must have
The sidebar menu enables the user to logout.

**UCR18** Must have
The sidebar menu enables the user to open an 'about' screen.

**Sidebar Menu when not logged in**

**UCR19** Should have
The sidebar menu displays previously used accounts.

**UCR20** Should have
The sidebar menu enables the user to log in with a previously used account.

**UCR21** Must have
The sidebar menu enables the user to open a screen where they can log in with an account that is new to this device.

**UCR22** Must have
The sidebar menu enables the user to open a screen for registration.

**UCR23** Must have
The sidebar menu enables the user to open an 'about' screen.

### 3.1.2 Registration screen

**UCR24** Must have
The registration screen allows the user to register a new TravelMatch account using an email address and password.

**UCR25** Must have
The registration screen allows the user to register a new TravelMatch account using Facebook login.

**UCR26** Won't have
The registration screen allows the user to register via other authentication providers.

### 3.1.3 Login screen

**UCR27** Must have
The login screen allows the user to authenticate using an email address with the associated password.

**UCR28** Must have
The login screen allows the user to connect with Facebook.

**UCR29** Won't have
The login screen allows the user to authenticate via other authentication providers.

### 3.1.4 User details screen

**UCR30** Must have
When logged in, the user can manage his account information in the user details screen.

**UCR31** Could have
The app supports multiple user accounts stored on the same device.

**UCR32** Could have
The accounts on the app are distinguished by the authentication method and authentication method's username.

**UCR33** Could have
Users that were previously logged in, but are not currently logged in, can log in without re-entering their password within one week of that user's last activity on that device.

**UCR34** Could have
Users that were previously logged in, but are not currently logged in, have to re-enter their password to log in after one week of that user's last activity on that device.

**UCR35** Should have
In the user details screen, the user can add and/or edit their name.

**UCR36** Should have
In the user details screen, the user can add and/or edit their gender.

**UCR37** Should have
In the user details screen, the user can add and/or edit their birthdate.

**UCR38** Should have
In the user details screen, the user is able to delete their account.

**UCR39** Should have
When a user deletes their account, all their user data is removed from the device.

**UCR40** Could have
When a user deletes their account, all their user data is removed from the servers on the next connection to the server.

Requirement 41 has been removed

Requirement 42 has been removed

Requirement 43 has been removed

Requirement 44 has been removed

Requirement 45 has been removed

### 3.1.5 About screen

**UCR46** Must have
The app includes an about screen that displays basic information about the app.

**UCR47** Must have
All required license information is displayed to users in the about screen.

**UCR48** Must have
The about screen supports a way to contact iLysian with feedback, comments or questions.

### 3.1.6 Vacation details

**UCR49** Must have
The vacation details screen contains an input field with flexibility range for the departure date.

**UCR50** Must have
The vacation details screen contains an input field with flexibility range for the return date.

**UCR51** Must have
The vacation details screen contains an input field for the budget per person.

**UCR52** Must have
The vacation details screen allows the user to indicate no budget preference.

**UCR53** Must have
The vacation details screen contains an input field for the number of adults.

**UCR54** Must have
The vacation details screen contains an input field for the number of children.

**UCR55** Should have
When a user is logged in, the vacation details screen by default displays the latest entered vacation details of that user on any device.

**UCR56** Should have
When a user is not logged in, the vacation details screen displays the latest entered vacation details of that user on that particular device.

**UCR57** Must have
If all the input fields contain valid values the user can start the interest analysis.

**UCR57a** Won't have
When the interest analysis is started the vacation details are automatically saved under the default name.

**UCR57b** Won't have
The vacation details screen contains an option to save the entered vacation details under a name chosen by the user.

**UCR57c** Won't have
The vacation details screen contains a menu for switching between previously saved vacation details using the chosen name.

**UCR57d** Won't have
If the user has no stored vacations the name field is automatically filled in with the translation of the words "vacation" in the app's language.

**UCR57e** Won't have
From the vacation details screen the user is able to delete previously saved vacation details.

**UCR58** Must have
The vacation details screen contains a header that allows the user to go to the sidebar menu.

### 3.1.7 Interest analysis

**UCR59** Must have
The interest analysis screen allows the user to rate one photo at a time.

**UCR60** Must have
Photos shown in the interest analysis are loaded from the back end server.

**UCR61** Must have
The user is able to "like" the photo.

**UCR62** Must have
The user is able to "dislike" the photo.

**UCR63** Must have
After a constant number of choices an animation is displayed telling the user the advice is being calculated.

**UCR64** Must have
The animation is displayed for a predefined minimum amount of time.

**UCR65** Must have
After the animation finishes the user receives a recommendation from the back end server.

**UCR66** Must have
The interest analysis screen contains a small header that can expand.

**UCR67** Must have
The interest analysis screen's header can be used to go back to the vacation details screen.

**UCR68** Must have
The interest analysis screen's header can be used to open the sidebar menu.

**UCR69** Must have
The interest analysis screen displays the progress of (dis)liking.

**UCR70** Must have
The interest analysis screen sends each image rating of the user to the back end server.

**UCR71** Must have
Upon receiving the recommendation from the back end server, the app displays this in the first advice hotel overview screen.

### 3.1.8 Hotel overview

**UCR72** Must have
The hotel overview screen contains an overview of hotels with the same destination.

Requirement 73 has been removed

**UCR74** Must have
The hotel overview screen supports scrolling.

**UCR75** Must have
For every hotel in the overview, the hotel overview screen contains an image of the hotel.

**UCR76** Must have
For every hotel in the overview, the hotel overview screen contains the total price that has to be paid per person if the user wants to book that vacation offer.

**UCR77** Could have
For every hotel in the overview, the hotel overview screen contains a user rating.

**UCR78** Should have
For every hotel in the overview, the hotel overview screen contains a Hotelstar rating.

**UCR79** Must have
For every hotel in the overview, the user is able to open a hotel details view.

**UCR80** Must have
The hotel overview screen contains a header with the name of the current recommended destination.

**UCR81** Must have
The hotel overview screen contains a header that allows you to go back to your vacation details.

**UCR82** Must have
The hotel overview screen contains a header that allows you to open the sidebar menu.

**UCR83** Should have
When a user saves a destination, they will receive some feedback.

**First advice hotel overview**

**UCR84** Must have
When the first advice hotel overview is shown, the hotel overview screen contains a header that allows the user to request a second advice based on the users *Travel DNA*.

Requirement 85 has been removed

**UCR86** Won't have
When the first advice hotel overview is shown and the user is logged in, the hotel overview screen enables the user to save the hotels with the current destination.

**Second advice hotel overview**

Requirement 87 has been removed

**UCR88** Won't have
When the second advice hotel overview is shown and the user is logged in, the hotel overview screen enables the user to save the hotels with the current destination.

**UCR89** Must have
When the second advice hotel overview is shown, the hotel overview screen contains a header that allows the user to go back to the first advice.

**Previously saved hotel overview**

**UCR90** Won't have
When the previously saved hotel overview is shown, it displays the same hotels as when user chose to save the destination.

**UCR91** Won't have
When the previously saved hotel overview is shown, it must be visible to the user when certain hotels are not available anymore.

**UCR92** Won't have
When the previously saved hotel overview is shown, the user must be able to refresh from the back end server.

**UCR93** Won't have
When a user refreshes from a previously saved hotel overview a new set of hotels is obtained from the back end server.

**UCR94** Won't have
When a user has refreshed the hotel over screen the hotel overview screen enables the user to save the hotels with the current destination.

**UCR95** Won't have
When a user saves a destination that was already saved earlier, the set of hotels saved for that destination are overwritten by the new set.

### 3.1.9 Hotel details

**UCR96** Must have
The hotel details are shown inside the hotel overview screen.

**UCR97** Must have
The hotel details contain an image of the hotel.

**UCR98** Must have
The hotel details contain the name of the hotel.

**UCR99** Must have
The hotel details contain the total price that has to be paid per person if the user wants to book the vacation.

**UCR100** Must have
The hotel details contain a description of the hotel.

**UCR101** Should have
The hotel details contain an average user rating of the hotel if it has one.

**UCR102** Should have
The hotel details contain the Hotelstar rating for the hotel.

**UCR103** Won't have
From the hotel details view the user is able to book within the app.

**UCR104** Must have
From the hotel details view the user is able to book by being redirected to the relevant web page of the travel agency in the phone's browser.

### 3.1.10 Back end

**Images**

**UCR105** Must have
Images used for interest analysis are stored on a server.

**Travel DNA**

**UCR106** Must have
Each user can have a *Travel DNA*, which is stored on a back end server.

Requirement 107 has been removed

**UCR108** Won't have
Each user is able to reset their *Travel DNA* to obtain different recommendations.

**UCR109** Won't have
Each user only has one *Travel DNA* stored for each of their previously saved vacation details.

**UCR109a** Won't have
*Travel DNA* of different vacation details do not interfere.

**UCR110** Must have
Images used for interest analysis have a value for each related *DNA attribute* stored on a server.

**UCR111** Must have
*Travel DNA* consists of stored (dis)likes for a set of images.

**CMS**

**UCR112** Must have
The CMS can be accessed by assigned managers.

Requirement 113 has been removed

Requirement 114 has been removed

**UCR115** Won't have
The CMS is able to change the amount of images that have to be liked or disliked before the recommendation is given.

**UCR116** Could have
For changes in the database that could affect the recommendation that is given to end users, the CMS has an draft functionality before publication of these changes.

**UCR117** Could have
When a change is in draft the change is not applied yet.

**UCR118** Could have
Multiple changes that are in draft can be applied in one interaction with the CMS.

**UCR119** Must have
The CMS allows to create new *DNA attributes*.

**UCR120** Must have
Each destination offered by TravelMatch has some value for each related *destination attribute*.

**UCR121** Must have
When a new tag is created it can be saved as draft until its value has been specified for all locations.

**UCR122** Could have
The CMS provides functionality to delete image tags.

**UCR123** Could have
The CMS provides functionality to delete attributes when the value is zero for all photos in the database.

**UCR124** Must have
The CMS provides functionality to upload photos to the server.

**UCR125** Must have
The CMS provides functionality to delete photos from the server.

**UCR126** Must have
The CMS provides functionality to set values of attributes per photo.

**UCR127** Must have
The CMS provides functionality to change values of attributes per photo.

**UCR128** Must have
The CMS provides functionality to add destinations.

**UCR129** Must have
The CMS can add an initial value for each attribute of new destinations.

**UCR130** Won't have
The CMS allows certain hotels to be excluded from the hotel overview screen.

**UCR131** Could have
The CMS allows the assignment of a priority to each hotel which is used in selecting the hotels of the hotel overview screen.

**AI module**

**UCR132** Must have
The recommendation that is given consists of one destination and a variable number of hotels.

**UCR133** Must have
The recommended destination is the destination best matching the *Travel DNA*.

**UCR134** Won't have
If the analytics obtain information about what the user thinks of a destination this information is used as feedback to the *destination attributes*.

**UCR135** Won't have
The (dis)liking of previous images influences the sequence of following images still to be judged.

**UCR135a** Must have
The set of images shown until now influences the sequence of following images still to be judged.

**UCR136** Should have
The images presented to the user are selected to give the most information gain of the user's *Travel DNA*.

**UCR137** Must have
The recommended hotels are always located in or near the recommended destination.

**UCR138** Should have
The recommended hotels for each destination are stored in a database, cached from the Arke feed.

**UCR139** Could have
The first set of recommended hotels contains the cheapest hotel for that destination.

**UCR140** Could have
The first set of recommended hotels contains the hotel closest to the budget for that destination.

**UCR141** Could have
The first set of recommended hotels contains the hotel that is most profitable to TravelMatch for that destination.

**UCR142** Could have
The first set of recommended hotels contains the hotel with the highest user rating for that destination.

**UCR143** Won't have
The first set of recommended hotels contains the hotel with the highest priority assigned in the CMS for that destination.

### 3.1.11 Analytics

**UCR144** Must have
The app sends analytics events to the analytics provider on every analytics event.

**UCR145** Must have
An event that is recorded contains its timestamp.

**UCR146** Must have
If a user is logged in, an event contains the user account identifier.

**UCR147** Must have
If a user is not logged in, an event contains the user's session.

**UCR148** Must have
If a user is not logged in, an event contains the user's device id.

**UCR149** Could have
If a user is logged in, information about the phone used is recorded on every analytics event.

**UCR150** Must have
The data that is entered on the vacation details screen must be recorded when a person starts the interest analysis.

**UCR151** Must have
The event of (dis)liking of every image must be recorded.

**UCR152** Won't have
The event of saving a destination for later must be recorded.

**UCR153** Must have
The event of clicking on a hotel must be recorded.

**UCR154** Must have
The event of asking for a second advice must be recorded.

**UCR155** Must have
The event of going back to the vacation details screen must be recorded.

## 3.2 Constraint requirements

### 3.2.1 App environment

**UCR156** Must have
The TravelMatch app can run on smart-phone devices running Android 4.1 "Jelly Bean" or newer.

**UCR157** Must have
The TravelMatch app can run on tablet devices running Android 4.1 "Jelly Bean" or newer.

**UCR158** Must have
The TravelMatch app can run on smart-phone devices running iOS 7.0 or newer.

**UCR159** Must have
The TravelMatch app can run on tablet devices running iOS 7.0 or newer.

**UCR160** Could have
The TravelMatch app is available for free through the Google Play Store for all supported Android devices.

**UCR161** Could have
The TravelMatch app is available for free through the App Store for all supported iOS devices.

**UCR162** Won't have
The TravelMatch app is supported on Android Wear.

**UCR163** Won't have
The TravelMatch app is supported on Android TV.

**UCR164** Won't have
The TravelMatch app is supported on Apple TV.

**UCR165** Won't have
The TravelMatch app is supported on Apple Watch.

**UCR166** Won't have
The TravelMatch app is supported on Windows Phone.

**UCR167** Won't have
The TravelMatch app is supported on Desktop or Laptop OS's.

### 3.2.2 Adaptability

**UCR168** Must have
TravelMatch should be designed to make it easy to add other affiliate feeds.

---

**UCR169** Won't have
TravelMatch should be designed to make it easy to switch to a *Waverunner* like system.

---

**UCR170** Must have
TravelMatch should be designed with an in-app booking functionality in mind.

---

**UCR171** Must have
TravelMatch should be designed to make it easy to switch to another user authentication mechanism.

---

### 3.2.3 Resources

**UCR172** Must have
The TravelMatch app is written using the Ionic framework.[20]

---

### 3.2.4 Licensing

**UCR173** Should have
TravelMatch is not written under any license agreement at this time.

---

### 3.2.5 Performance Requirements

**UCR174** Could have
Interaction with the TravelMatch app provide some visual feedback within 0.25 seconds on any of the supported devices.

---

**UCR175** Should have
Interaction with the TravelMatch app provide some visual feedback within 1 seconds on any of the supported devices.

---

**UCR176** Must have
Interaction with the TravelMatch app provide some visual feedback within 2 seconds on any of the supported devices.

---

**UCR177** Should have
Interaction with the CMS provide some visual feedback within 5 seconds.

---

**UCR178** Must have
Interaction with the CMS provide some visual feedback within 15 seconds.

---

**UCR179** Should have
The recommendation screen appears in less then 4 seconds after the last image has been swiped.

---

**UCR180** Must have
The recommendation screen appears in less then 5 seconds after the last image has been swiped.

**UCR181** Must have
The database supports at least 1000 images.

**UCR182** Should have
The database supports at least 10000 images.

**UCR183** Must have
The database supports at least 100 tags.

**UCR184** Should have
The database supports at least 500 tags.

**UCR185** Must have
The database supports at least 100000 users including guest users.

**UCR186** Should have
The database supports at least 500000 users including guest users.

**UCR187** Must have
The database supports at least 500 analytics event records per user.

**UCR188** Must have
The database supports at least 200 different destinations.

**UCR189** Should have
The database supports at least 1500 different destinations.

## 3.3 Changes in user requirements

Some user requirements have been changed after the document was approved, in agreement with the client. Below a list of modified, edited and removed user requirements are given. In following documents all changed requirements will be used accordingly. User requirements that are removed are not required anymore by the client. An explanation of each change is given on the following page.

| **User requirement** | **change status** |
|---|---|
| UCR2 | modified |
| UCR2a-d | added |
| UCR41 | removed |
| UCR42 | removed |
| UCR43 | removed |
| UCR44 | removed |
| UCR45 | removed |
| UCR49 | modified |
| UCR50 | modified |
| UCR55 | modified |
| UCR56 | modified |
| UCR57a-e | added |
| UCR73 | removed |
| UCR85 | removed |
| UCR86 | modified |
| UCR87 | removed |
| UCR88 | modified |
| UCR90 | modified |
| UCR91 | modified |
| UCR92 | modified |
| UCR93 | modified |
| UCR94 | modified |
| UCR95 | modified |
| UCR96 | modified |
| UCR107 | removed |
| UCR109 | modified |
| UCR109a | added |
| UCR113 | removed |
| UCR114 | removed |
| UCR115 | modified |
| UCR134 | modified |
| UCR135 | modified |
| UCR121 | modified |
| UCR129 | modified |
| UCR134 | modified |
| UCR135 | modified |
| UCR135a | added |
| UCR149 | modified |
| UCR152 | modified |
| UCR179 | modified |

UCR2 and added UCR2a-d are added and modified to give more introduction to the application. The starting vacation details screen was not sufficient to introduce the app to users.
UCR41-45 were removed as we discussed with the client the implications of using sessions.
UCR56 and 57a-e were modified and added to describe the vacation details saving that can be added after this project.
UCR73 is removed because a correctly implemented direct booking system will never display a location for which no matching hotels can be displayed.
UCR107 is removed to be rewritten as a sub requirement UCR109a to be more clear and according to the needs of the customer.
UCRs 85, 87, 113 and 114 were removed due to changes in the application design.
UCRs 86, 88, 90-95, 109(a), 115, 134, 135 and 152 were removed from the scope of this project.
UCR135a was added as a new method in which entropy is calculated.
UCR49 and UCR50 now include the flexible departure and return date ranges.
UCR96 was modified to reflect the new way in which vacation details are shown.
UCR121 was rewritten to be more clear.
UCR129 was rewritten to be more clear.
UCR149 was modified to record more analytics information.
UCR179 was prolonged after the longer calculating animation was implemented.

# Appendix A

# Use cases

## A.1 General use cases

### A.1.1 Register an account via email

| | |
|---|---|
| **Goals:** | Register an account. |
| **Preconditions:** | The user has a valid email address and can think of a password. |
| **Summary:** | The user obtains an account by registering with email and password. |
| **Priority:** | Must have |
| **Steps:** | |

| *Actor actions:* | *TravelMatch response* |
|---|---|
| 1. Open the sidebar menu. | |
| | 2. Show sidebar menu drawer. |
| 3. Choose to register a new account. | |
| | 4. Ask user to connect with Facebook or register a new account via email/password. |
| 5. Choose to register a new account via email/password. | |
| | 6. Show email address and password input fields. |
| 7. Input email address and desired password. | |
| | 8. Register account on back end server with pending activation. |
| | 9. Send a verification email to the entered email address. |
| 10. Click the verification link in the email. | |
| | 11. Change user status to activated. |
| | 12. Remove pending activation. |
| 13. Reopen TravelMatch and login. | |

**Alternatives:**

1. In step 3, if a user is already logged into the application, the user must choose to switch accounts instead. Then the user can choose to register a new account.
2. In step 9, if someone already tried to register that email address but the address's owner did not confirm it, a new verification mail will be sent.

### A.1.2 Log into an account via email

| | |
|---|---|
| **Goals:** | Log into an account. |
| **Preconditions:** | The user previously registered an account with the same email address. |
| **Summary:** | The user logs into an account by registering with email and password. |
| **Priority:** | Must have |
| **Steps:** | |

| *Actor actions:* | *TravelMatch response* |
|---|---|
| 1. Open the menu. | |
| | 2. Show menu drawer. |
| 3. Choose to log into an account. | |
| | 4. Show email address and password input fields. Ask user to log in via email/password or connect with Facebook. |
| 5. Input email address and password and click login. | |
| | 6. Confirm that the entered email address and password match an account in the database. |
| | 7. Send user authentication token. |
| 8. Store user authentication token in local storage. | |

**Alternatives:**

1. In step 3, if a user is already logged into the application, the user must choose to switch accounts instead.
2. In step 6, if the entered email address and password do not match any account in the database, the app informs the user of this and then returns to step 5.

### A.1.3 Register and log into an account via Facebook

**Goals:** Log into an account.
**Preconditions:** The user has a Facebook account.
**Summary:** The user logs into an account by registering via Facebook.
**Priority:** Must have
**Steps:**

| *Actor actions:* | *TravelMatch response* |
|---|---|
| 1. Open the menu. | |
| | 2. Show menu drawer. |
| 3. Go into account settings. | |
| | 4. Show connect with Facebook button. |
| 5. Choose to connect with Facebook. | |
| | 6. Redirect user to Facebook authorization. |
| 7. Sign in to Facebook account. | |
| 8. Authorize TravelMatch to use Facebook account. | |
| | 9. Register account on back end server. |
| | 10. Send user authentication token. |
| 11. Store user authentication token in local storage. | |

**Alternatives:**

1. In step 3, if a user is already logged into the application, the user must choose to log out instead.
2. In step 7, if the user is already logged into their Facebook account, this step may be skipped.
3. In step 9, if the user had previously logged into the TravelMatch account with the same Facebook account, this step may be skipped.

## A.2 Specific use cases

### A.2.1 Trying out without creating an account

**Goals:** Try out the app for fun, to see how it works.
**Preconditions:** The user is not logged in.
**Summary:** The user obtains a destination advice.
**Priority:** Must have
**Steps:**

| *Actor actions:* | *TravelMatch response* |
|---|---|
| 1. Fill in vacation details. | |
| 2. Choose to start advice. | |
| | 3. Deliver set of images to show. |
| 4. Like or dislike all images. | |
| | 5. Update *Travel DNA*. |
| | 6. Determine destination advice. |
| | 7. Determine available hotels. |
| | 8. Send destination advice and hotels. |
| 9. Choose a hotel. | |
| | 10. Display hotel details. |

**Alternatives:**

1. In step 9, the user may instead choose to receive a second advice, upon which the second destination and set of available hotels is shown.

### A.2.2 Booking a vacation on a new device

**Goals:** Book a vacation.
**Preconditions:** The user has an account.
**Summary:** The user logs into their account on a new device, receives a destination advice and books a vacation.
**Priority:** Must have
**Steps:**

| *Actor actions:* | *TravelMatch response* |
|---|---|
| 1. Log into account (see use cases A.1.2 and A.1.3). | |
| 2. Fill in vacation details. | |
| 3. Choose to start advice. | |
| | 4. Deliver set of images to show. |
| 5. Like or dislike all images. | |
| | 6. Update *Travel DNA*. |
| | 7. Determine destination advice. |
| | 8. Determine available hotels. |
| | 9. Send destination advice and hotels. |
| 10. Choose a hotel. | |
| | 11. Display hotel details. |
| 12. Choose to book the hotel. | |
| | 13. Redirect user to website of travel agency.. |

**Alternatives:**

1. In step 10, the user may instead choose to receive a second advice, upon which the second destination and set of available hotels is shown.